THREE-DIMENSIONAL TSUNAMI MODELING USING GPU-SPHYSICS

A Senior Scholars Thesis

by

ANDREW JAMES MUNOZ

Submitted to the Office of Undergraduate Research of
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLAR

April 2010

Major: Geophysics

THREE-DIMENSIONAL TSUNAMI MODELING USING GPU-SPHYSICS

A Senior Scholars Thesis

by

ANDREW JAMES MUNOZ

Submitted to the Office of Undergraduate Research of
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLAR

Approved by:

Research Advisor:	Robert Weiss
Associate Dean for Undergraduate Research:	Robert C. Webb

April 2010

Major: Geophysics

ABSTRACT

Three-Dimensional Tsunami Modeling Using GPU-SPHysics. (April 2010)

Andrew James Munoz
Department of Geology and Geophysics
Texas A&M University

Research Advisor: Dr. Robert Weiss
Department of Geology and Geophysics

With the devastating effects of the 2004 Sumatra Tsunami, tsunami research is at an all time high. Tsunami forecasting and modeling has become exceedingly important in the anticipation of major disasters. Tsunami inundation modeling, how a tsunami invades a coastal area, is an extremely useful tool for the prevention of major disaster in tsunami laden zones. Using a new free-surface hydrodynamic modeling code called GPU-SPHysics, accurate inundation and propagation models of tsunamis can be modeled at very high resolutions. GPU-SPHysics takes advantage of the extremely powerful computational power of a GPU (Graphics Processing Unit) and calculates the dynamics of fluids based on SPH (Smoothed Particle Hydrodynamics). The implementation of SPH on the GPU not only creates accurate, three-dimensional models but stunning visualizations of a tsunami wave breaking on beaches or other structures. Using the data from these models, coastal communities will be well prepared for any magnitude of tsunami that they may encounter by adjusting their infrastructure and disaster preparation to accommodate for this common disaster and potentially save many lives. To utilize GPU-SPHysics' models accurately, they must first be verified. NOAA (The National Oceanic and Atmospheric Administration) has provided benchmarks for tsunami inundation and propagation models. These benchmarks consist of analytic tests, laboratory tests and field tests. A key benchmark for GPU-SPHysics to be verified against is the solitary wave inundation on a sloping beach

experiment. The solitary wave best represents the leading wave of a tsunami; hence it is vital to test other inundations that involve more complex structures than sloping beaches. Through visual analysis, the GPU-SPHysics solitary wave model, accurate to a small deviation, has been verified using the analytic calculation for maximum runup as provided by Synolakis. To verify other benchmarks provided by NOAA, GPU-SPHysics must be tested against multiple experiments. Once GPU-SPHysics has been verified for multiple data sets, it can be considered an accurate tool for hazard analysis.

ACKNOWLEDGMENTS

Thank you to Dr. Robert Darlymple for allowing us to use GPU-SPHysics and to Dr. Robert Weiss for providing me with this project and great mentoring.

NOMENCLATURE

CUDA	Compute Unified Device Architecture
DEM	Digital Elevation Map
GPU	Graphics Processing Unit
NOAA	National Oceanic and Atmospheric Administration
SPH	Smoothed Particle Hydrodynamics

TABLE OF CONTENTS

Page

ABSTRACT iii

ACKNOWLEDGMENTS v

NOMENCLATURE vi

TABLE OF CONTENTS vii

LIST OF FIGURES ix

CHAPTER

I INTRODUCTION 1

Fluid dynamics 2

Euler equations 4

Navier-Stokes equation 5

The solitary wave and N-wave 7

Smoothed-particle hydrodynamics 8

GPU-SPHysics 10

Verification and validation of models 13

II METHODS 16

Hardware 16

Software 17

GPU-SPHysics structure 18

The solitary wave experiment 19

The conical island experiment 20

III RESULTS 22

Data acquisition 22

Solitary wave results 22

CHAPTER Page

Conical island results 24

IV ANALYSIS AND SUMMARY 26

Conclusions 26

REFERENCES 28

CONTACT INFORMATION 30

LIST OF FIGURES

FIGURE		Page
1	Stress cube	3
2	CPU to GPU speedup	11
3	GPU architecture	12
4	Canonical bathymetry sketch	14
5	Tesla GPU	16
6	Island dimensions	21
7	Conical island laboratory runup	21
8	Runup vs. time	23
9	Solitary wave break	24
10	Conical island GPU-SPHysics maximum runup	25
11	Inundation length measurement method	27

CHAPTER I

INTRODUCTION

Tsunamis have proven to be a deadly hazard throughout history. Following the devastation of the 2004 Sumatra tsunami, tsunami modeling has become increasingly popular as method in predicting the threat of potential tsunami destruction. Modeling the propagation and inundation of a tsunami accurately would help prevent potentially deadly situations and save many lives. Coupled with advances in tsunami forecasting, tsunami inundation models are a powerful tool in assessing the environmental hazard caused by a tsunami. Since these models are essential in determining potentially deadly situations, they must be precise and accurate. Therefore the National Oceanic and Atmospheric Administration (NOAA) has developed a set of standards, criteria and procedures that models must follow to be considered verified and validated as a tsunami inundation model [Synolakis, *et al*., 2007]. NOAA requires a rigorous verification and validation process so the models can be relied upon to save human lives. Benchmark tests for verification and validation given by NOAA include both analytical tests as well as laboratory tests. Each test focuses primarily on the runup of a tsunami as it is the most dangerous hazard associated with a tsunami [Briggs, *et al*, 1995]. The runup is how far up the land a wave travels vertically, so any nearby structures or people in the path of the runup of a tsunami are in danger. The runup of a modeled tsunami wave onto different geometries must be accurate in every aspect so that the model is reliable in the prevention of damage from a tsunami at any coastal location.

This thesis follows the style of the journal of Coastal Engineering.

Fluid dynamics

Fluid dynamics studies the motion of fluids when forces such as pressure and gravity are applied. Models of fluid motion calculate and observe individual fluid particles. Modeled particles are described by their position, velocity, acceleration and mass. These basic parameters are used to model particle forces and dynamics. Force is defined using Newtons second law of motion which states that:

$$\mathbf{F} = m\mathbf{a} \tag{1}$$

where $\mathbf{F}$ is a force, m is the mass of the object which the force is applied to and $\mathbf{a}$ is the acceleration of the object. A boldface variable represents a vector. Acceleration is defined as:

$$\mathbf{a}(t) = \frac{d\mathbf{v}(t)}{dt} \tag{2}$$

which is the rate of change of $\mathbf{v}(t)$, the velocity of a particle, over time. The velocity is defined as:

$$\mathbf{v}(t) = \frac{d\mathbf{x}(t)}{dt} \tag{3}$$

which is the rate of change of $\mathbf{x}(t)$, the position of a particle, over time. The velocity is also represented as a vector function

$$\mathbf{v}(x,y,z,t) = \mathbf{i}u(x,y,z,t) + \mathbf{j}v(x,y,z,t) + \mathbf{k}w(x,y,z,t) \tag{4}$$

where $\mathbf{i}$, $\mathbf{i}$, $\mathbf{i}$ are unit vectors and the scalars u, v, w are each a scalar field [White, 2003]. Force, velocity and acceleration, are further applied to conservation equations. Fluids, liquids and gasses, are constantly in motion by stresses. Shear stress is a force applied to a particle tangential to the area over which it is applied. However, normal stress is the force applied to a particle perpendicular to the area over which is applied.

Normal stress is also called pressure. The pressure in a fluid is caused by the normal stress

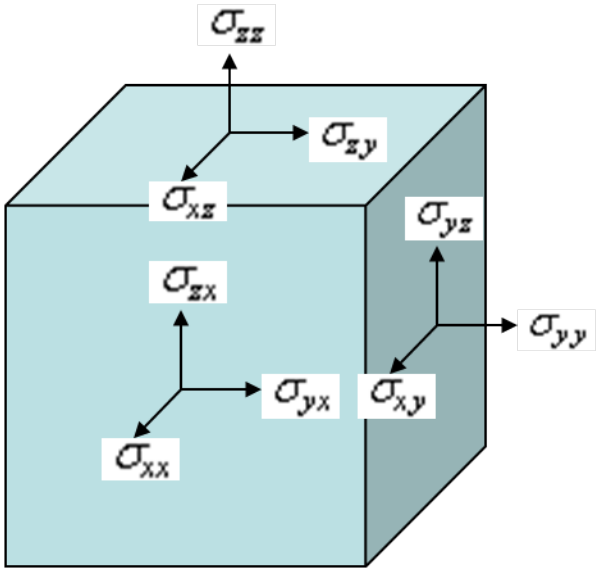

Fig. 1. Stress cube. Net force of stresses applied to an arbitrary body

applied to any side of a particle. If these stresses are equal, then the pressure is then called hydrostatic pressure. Hydrostatic pressure occurs when the stresses on a body occur from each direction and are equivalent as shown by figure 1 and equation 5.

$$\sigma_{xx} = \sigma_{yy} = \sigma_{zz} = \sigma_{xy} = \sigma_{xz} = \sigma_{yz} \tag{5}$$

As water depth increases in a body, the hydrostatic pressure increases as a result due to an increase in the overlying fluid. Density is defined as

$$\rho = \frac{m}{V} \tag{6}$$

where ρ is density, m is mass and V is the volume. Pressure can be directly related to density. For a gas, when a pressure is applied, the gas decreases in volume greatly, while when the same pressure is applied to a liquid of the same initial volume, it only slightly decreases in volume. The compressibility of the gas is much greater than that of the liquid; hence gas is a compressible fluid while liquid is mostly an incompressible fluid. However, for complex velocity fields in three-dimensional particle systems, the weak compressibility of water must be accounted for. For shear stresses, the fluid undergoes deformation which

is called strain. Viscosity is the quantitative measure of a fluid's resistance to flow [White, 2003]. Viscosity is useful since it determines the fluid strain rate given by an applied shear stress [White, 2003]. Shear is proportional to strain rate, and since strain rate is equal to the velocity gradient of a fluid (by geometry) then

$$\tau = \mu \frac{\partial \mathbf{v}}{\partial y} \tag{7}$$

where τ is the applied shear, is the viscosity term, and $\frac{\partial \mathbf{v}}{\partial y}$ is the velocity gradient for common linear fluids [White, 2003]. Linear fluids that follow equation 7 are called Newtonian Fluids [White, 2003]. This law directly indicates that with a given shear stress, fluid is set into motion.

Euler equations

Euler equations describe the fluid of an inviscid flow of a compressible fluid [Ward, 2003]. An inviscid flow has zero viscosity. They are used in classical tsunami modeling and to derive the more commonly used Navier-Stokes Equation. Euler equations are essentially three laws of conservations for fluids, the conservation of mass, the conservation of momentum, and the conservation of energy. The law of the conservation of mass derives from the first law of thermodynamics and it states that mass cannot be created or destroyed [Cap, 2006]. For a fluid system, mass must be conserved. The net flux must be zero for the entire system to uphold conservation of mass. A form of the conservation of mass equation from [Haberman and John, 2006] is

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{8}$$

where $\rho(x,y,z,t)$ is density and $\mathbf{v}(x,y,z,t)$ is the velocity of a fluid. Equation 8 is also known as the continuity equation. If the fluid is incompressible, then the density is not a function of pressure, so

$$\nabla \cdot \mathbf{v} = 0 \tag{9}$$

Hence, volume is conserved in the system. The conservation of mass is a basic principle used in fluid dynamics and is a key step in deriving the equations describing fluid systems. Once conservation of mass is checked, conservation of momentum must also be validated. Momentum can also be derived from Newton's second law of motion by seeing that the net force, the sum of all forces in a system, is equal to

$$\mathbf{F}_{net} = \frac{d(m\mathbf{v})}{dt} \tag{10}$$

The conservation of momentum is then derived from [Haberman and John, 2006] as

$$\frac{\partial \mathbf{v}}{dt} + \frac{\nabla P}{\rho} = 0 \tag{11}$$

where P is pressure. The conservation of mass is used to simplify the conservation of momentum to the conservative form in equation 9. The conservation of energy is derived from [Haberman and John, 2006] as

$$\frac{\partial E}{\partial t} + \left(\frac{P}{\rho}\right) \nabla \cdot \mathbf{v} = 0 \tag{12}$$

where E is the energy. These three conservation equations form the Euler equations of an inviscid flow for a compressible fluid.

Navier-Stokes equation

The Navier-Stokes equation is one of the most influential equations in the history of physics. Using the Euler equations and an added viscosity term due to the account of shear stress in a fluid, the Navier-Stokes equation for incompressible fluids is

$$\frac{\partial \mathbf{v}}{\partial t} + \nabla \cdot \mathbf{v} = g - \frac{\nabla P}{\rho} + \frac{\mu}{\rho} \nabla^2 \mathbf{v} \tag{13}$$

where ρ is the density, μ is the viscosity coefficient, v is the velocity from equation 4 and g is the gravitational constant 9.81 m/s^2. The Navier-Stokes equations are used to solve for the pressure, density and velocity components of a flow, although in this instance, the density ρ is held constant for the fluid. Each term has a special significance. First, the g term

describes the gravity force in the z direction and the coriolis, rotational, and other body forces in the x and y direction. ∇P is the term describes the pressure gradient which must be balanced by a net force of gravity, acceleration, or some other effect [White, 2003]. Since the fluid is incompressible, the viscosity term, $\mu\nabla^2\mathbf{v}$ describes a constant viscosity throughout the fluid. If the fluid were compressible, then an additional viscosity term would be added to the equation to account for compressibility, but since $\nabla \cdot \mathbf{v} = 0$ for incompressible fluids, the term disappears. Describing the local acceleration of the fluid, the $\frac{\partial \mathbf{v}}{\partial t}$ term shows the acceleration of the fluid temporally. The term which describes advective acceleration is the $\nabla \cdot \mathbf{v}$ term which shows the spatially varying velocity of the fluid.

In a more complex and violent flow, turbulence may arise. Turbulence is the random and disorderly small fluctuations in velocities caused by the destabilization of viscosity [White, 2003]. Turbulence is mathematically determined to exist in a fluid by using a dimensionless number called the Reynolds number. The Reynolds number is high when the fluid is turbulent and it is low when the fluid is laminar. A term describing turbulence must be added to the Navier-Stokes equations in order to properly describe the high-Reynolds number portions of a compressible fluid.

To further simplify the Navier-Stokes equation in certain situations, the assumption of zero angular velocity, or irrotationality, is used [White, 2003]. It can be shown that the angular velocity ω is related to vorticity ζ and velocity $\mathbf{v}$ by

$$\zeta = 2\omega = (curl\mathbf{v}) \tag{14}$$

where vorticity is the tendency for a particle to rotate [White, 2003]. For irrotational flows with zero vorticity

$$curl\mathbf{v} \equiv 0 \tag{15}$$

This simplification is applied to the Navier-Stokes equations to obtain simplified solutions for low-Reynolds number fluids. The flow cannot be irrotational if there are significant

viscous forces influencing the fluid, density gradients exist caused by stratification, and if noninertial effects such as Coriolis acceleration from the earths rotation exist.

The solitary wave and N-wave

Tsunamis, due to their complex and variable formation that depends on their source, have constantly been a challenge for researchers to accurately model. Tsunamis caused by an earthquake in the deep ocean typically has a low wave height and a very long wavelength [Madsen, *et al*., 2008]. The evolution of tsunamis from a constant depth ocean to a variable depth shore causes the wave to vary in properties such as wave height, velocity, and period. The evolution of tsunamis has been shown, by many scientists, to be similar to the evolution of a solitary wave [Madsen, *et al*., 2008]. The solitary wave was first noted in 1884 by John Scott Russell who was studying the design of canal boats [Eilbeck, 2007]. The solitary wave was described as an autonomous unit which acted in many ways like a particle [Eilbeck, 2007]. The solitary wave theory, also known as soliton theory, was first derived from the KdV (Korteweg-de Vries) equation, which is a non-linear equation, and a soliton is a set of permanently propagating waves that decay at infinity [Eilbeck, 2007]. A solitary wave is a single wave that breaks from a soliton [Eilbeck, 2007]. The equation for a solitary wave at a constant depth is an exact solution to the KdV equation and is shown as

$$\eta(x,t) = hsech^2\left(\sqrt{\frac{3h}{4H^3}}(x-Ct)\right) \tag{16}$$

where η is the free-surface elevation of the wave, H is the water depth, h is the wave height, C is the celerity (velocity) of the wave, and (x,t) are the position and time coordinates of the wave [Synolakis, *et al*., 2008]. As shown, the free-surface elevation of the wave only depends on water depth and maximum wave height for a solitary wave propagating in a region of constant depth. Synolakis also developed analytical solutions to the initial value

problem of the breaking of solitary waves on various shaped shorelines of variable depth [Madsen, *et al*., 2008]. However, from the observations of many tsunami leading waves that often cause the water to recede greatly before the first wave arrives, (Flores Island in Indonesia 1992; at the Pacific coast of Nicaragua 1992; at Okushiri, Japan in 1993; East Java, Indonesia in 1994; Mindoro, Philippines in 1994; Manzanillo, Mexico in 1995; Chimboto, Peru in 1996 and now recently in Thailand 2004) scientists determined that a tsunami wave arrived onshore in two phases; either as a depression or an elevation followed by the opposite [Madsen, *et al*., 2008]. These waves are called leading depression N-waves (LDN) or leading elevation N-waves (LEN) [Madsen, *et al*., 2008]. The generalized N-wave is

$$\eta(x,0) = \alpha H(x-x_2)sech^2(K_s(x-x_1)) \quad where \quad K_s = \frac{1}{h}\sqrt{\frac{3H}{4h}} \tag{17}$$

where $\alpha < 0$ so that H is the wave height [Madsen, *et al*., 2008]. According to Madsen, Hammack and Segur, in 1974, verified the LDN theory by creating N-waves in an experiment. Unfortunately due to the unknown energy exchange between deep ocean earthquakes and the water column, tsunamis as N-waves are difficult to replicate accurately. Hence, solitary waves are an accurate and easy wave to replicate the propagation and inundation of tsunamis in a model since the analytical solutions are known. The implementation of a solitary wave into a working model will be explained in the next chapter.

Smoothed-particle hydrodynamics

Smoothed-Particle Hydrodynamics (SPH) was invented by [Gingold and Monaghan, 1977] as a method of applying conservation of mass, momentum and energy to particle dynamics. The assignment of mass, velocity and in some cases temperature, are known at any given point for a particle [Monaghan, 1988]. Particles are moved during what is called a time step which is a set interval of time that is able to properly calculate the parameters

of all particles at any given location. Since this method involves particles, it is considered Lagrangian. Each particle is free to move throughout the fluid, so there is no mesh. The mesh equivalent is the fluid particles moving with the fluid flow [Dalrymple and Rodgers, 1995b]. Hence, vorticity and turbulence are easily shown for a rotational flow [Dalrymple and Rodgers, 1995b]. The original application of SPH was to numerically model the fission of a rapidly rotating star [Monaghan, 1988]. Monaghan [1988] states that the application of SPH is most useful in three-dimensions; hence the application later discussed is best used using an SPH model. SPH was further applied to free-surface hydrodynamic flows by implementing the weakly compressible Navier-Stokes Equations. The Navier-Stokes Equations are used in SPH to predict the trajectories and interactions of particles in a fluid [Dalrymple and Rodgers, 1995b]. In order to save valuable computational time, the particle interactions must be calculated locally since the distant particle interactions are very small and insignificant. This reduction in computations is achieved by using a kernel where the 3D kernel is

$$W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j) = \frac{15}{16\pi h^3}\left(\frac{|\mathbf{x}-\mathbf{x}_j|^2}{4h^2} - \frac{|\mathbf{x}-\mathbf{x}_j|}{h} - 1\right) \tag{18}$$

and h is the smoothing length of the kernel [Dalrymple and Rodgers, 1995b]. The kernel is used to find the value of a particle j at an arbitrary point $\mathbf{x}$ by using the sum

$$f(\mathbf{x}) = \sum_j f_j W(\mathbf{x}-\mathbf{x}_j)V_j \tag{19}$$

where f_j is the value f of the particle j and V_j is the volume of the particle j [Dalrymple and Rodgers, 1995b]. In order to solve for the Navier-Stokes Equations, the conservation of mass is needed in particle form where

$$\frac{\partial \rho_i}{\partial t} = \sum_j m_j(\mathbf{u}_i - \mathbf{u}_j)\nabla_i W_{ij} \tag{20}$$

and the conservation of momentum is needed in particle form where

$$\frac{\partial \mathbf{u}_i}{\partial t} = -\sum_j m_j\left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \prod_{ij}\right)\nabla_i W_{ij} + g \tag{21}$$

and $\mathbf{u}_j$ is the velocity of the particle, m_j is the mass of the particle, P_j is the pressure at the particle, $\prod_{ij}$ is a viscosity term, and ∇_i is the gradient with respect to the coordinates of the particle i [Dalrymple and Rodgers, 1995b]. Lastly, the equation of state used is

$$P = B\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \tag{22}$$

where $\gamma = 7$ and B is related to the speed of sound [Dalrymple and Rodgers, 1995b]. An equation of state is necessary in that is directly relates the pressure to the density therefore reducing extra computations for a solution.

The typical implementation of SPH involves mostly 2D models due to the heavy computational load, but some 3D applications do exist. Darlymple and Hérault created SPHysics, a code that implements SPH and is available for free online. Due to a lack of processing power in CPUs (Computer Processing Units), the number of particles that can be modeled in 3D is greatly restricted. However, new and exciting breakthroughs in hardware and software development have broken this restriction.

GPU-SPHysics

Computing power reached an all-time high when the ability to use the graphics processor for computations was first used. As seen in Figure 2, between February and March of 2007, there was an order of magnitude increase in gigaflops, or how many billions of floating point calculations a processor can perform each second, for N-body calculations from using a GPU rather than a CPU (Computer Processing Unit). N-body simulations calculates the forces applied to each body of a set of constantly interacting bodies [Jorgensen *et al*., 2008]. This monumental increase in speed is a vital tool that will help produce much more accurate data and visually stunning models. As seen in the figure, the number of computations a GPU can handle is much greater than a CPU for N-body calculations.

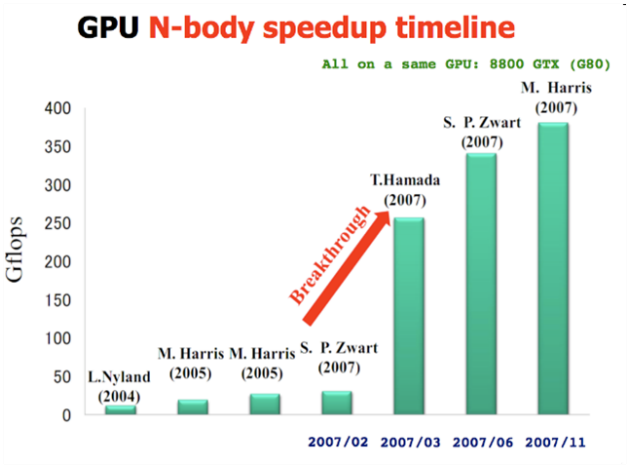

Fig. 2. CPU to GPU speedup. Notice an order of magnitude increase from 30 Gflops to 300 Gflops by using a GPU rather than a CPU for an N-body problem

NVIDIA, the graphics card company who invented the GPU, has developed a language to harness the use of the GPU for N-body calculations. CUDA (Compute Unified Device Architecture) is a computer language that serves as an extension to C. CUDA is a software platform that utilizes the massive number of processors, or threads, within a GPU in order to perform calculations at a much greater rate than a CPU [Halfhill, 2008]. Each thread is assigned a small task from the thread manager, which is build into the hardware and software that is provided by the GPU, and the task is then calculated by the threads [Halfhill, 2008]. CPUs only compute one instruction at a time for a non-paralized data set, but when many instructions need to be computed, this severely limits the speed a large set of data can be operated on individually, while in a GPU, the abundance of processors allows for many data sets to be operated on in a instinctively parallel way [Halfhill, 2008]. In addition, the data for the computation is stored on a local memory device shared by the threads, where in a CPU, the data is stored off the chip and must be accessed each time a task is assigned, so there is a small added time when the thread must be accessed off-chip hence the local storage of data is much more efficient [Halfhill, 2008]. A schematic diagram of the GPU architecture is shown in figure 3. A disadvantage to CUDA is the limitation NVIDIA has

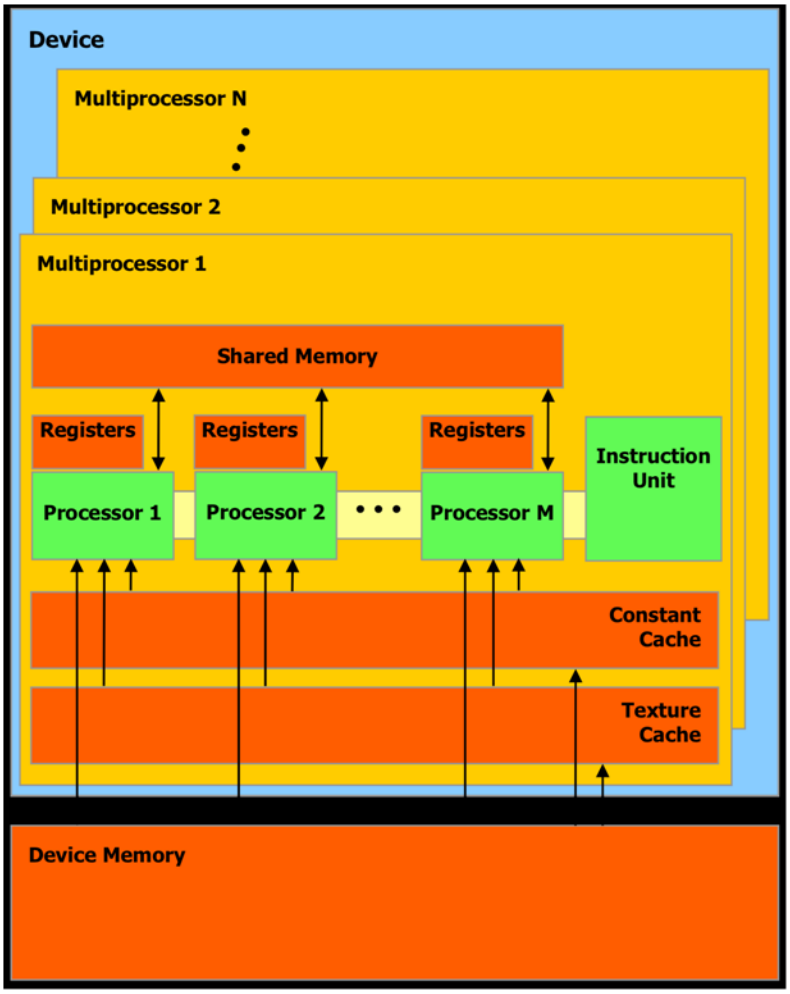

Fig. 3. GPU architecture. The organization of hardware within a NVIDIA GPU. The cache is simply a temporary storage area for data. [Hérault, *et al*., 2009]

put on the type of GPU that can be used. CUDA is specifically designed for a NVIDIA GPU only since the GPU has built-in hardware abstractions that CUDA uses as functions to control the GPU [Halfhill, 2008]. However, CUDA is able to work with C++, a high-level computing language, and OpenGL, a graphics creation tool, to create various useful applications for the GPU.

SPH modeling involves many particle calculations for force, pressure, density and neighbor searches, or particle interactions, based on the Navier-Stokes equations for each parti-

cle as described previously. Since particles must have the Navier-Stokes equations applied to them individually, then a high volume of computations is necessary to model a high-resolution fluid. The GPU was realized to have the ability to handle the massive computational load required for SPH by Kolb and Kuntz [Kolb and Cuntz, 2005]. Kolb and Kuntz's work however was done before CUDA was released, and instead they used OpenGL to program the GPU [Dalrymple and Hérault, 2009]. OpenGL is still used now in order to produce visualizations and an environment for the fluid to flow, but has no part in particle trajectory calculations. Hérault designed a new program called GPU-SPHysics which took the SPHysics code previously created and ran it on the GPU using CUDA. GPU-SPHysics was originally created by Hérault to model volcanic lava flows [Dalrymple and Hérault, 2009]. The code was further applied to free-surface hydrodynamics by Darlymple and Hérault [Dalrymple and Hérault, 2009]. GPU-SPHysics is broken into multiple classes which will be explained in detail in Chapter II. With the potential for running accurate and high-resolution simulations of real experiments, GPU-SPHysics is a valuable tool for modeling tsunamis.

Verification and validation of models

NOAA produced a set of standards and procedures that must be followed strictly so that a model can be used either as a tsunami forecast model or a tsunami runup model. NOAA expects to complete a fully standardized and coordinated tsunami hazard and risk assessment for all of the coastal regions of the United States and its territories, so various types of tsunami inundation models are needed [Synolakis, *et al*., 2007]. Rigorous analytical, laboratory and field benchmarks must be met so that the model accuracy is accurately validated (the process of ensuring that a model solves the parent equations of motion accurately) and verified (the process of ensuring that the model represents geophysical reality appropriately) [Synolakis, *et al*., 2007].

The first fundamental considerations, but most important, is that a model must obey mass conservation and convergence. Mass conservation for a tsunami model must be checked by showing that no volume of water was lost or added during the simulation. This can be accomplished by showing that

$$V(t) = \int_{X_{max}}^{X_S} \int_{Y_{max}}^{Y_S} \eta(x,y,t)dxdy \tag{23}$$

where V is the total displaced volume, η is the disturbed water depth, and the range is from the source region, X_S and Y_S, to the outer location, X_{max} and Y_{max} [Synolakis, *et al.*, 2007]. The convergence of a solution shows that the model is not unrealistic as extreme runups and rundowns are tested in the model [Synolakis, *et al.*, 2007]. Mass conservation and convergence are implicitly defined in the governing SPH equations.

Analytical verification checks the accuracy of the governing equations of the model. A basic verification is the inundation of a tsunami on a beach with canonical bathymetry (that is a beach with a constant depth region and a flat, sloping beach portion as seen in figure 4). Once a solitary wave runup is analytically verified on the simple beach problem, then

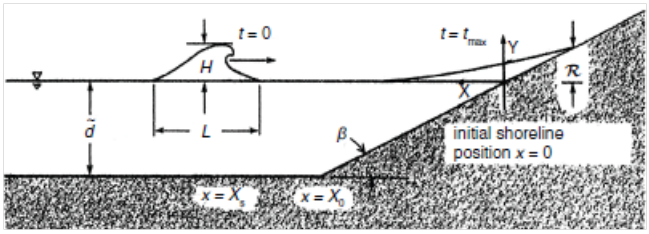

Fig. 4. Canonical bathymetry sketch. A canonical bathymetry sketch where H is the maximum wave height, d is the depth of the constant-depth portion, L is the wavelength of the wave, β is the sloping beach angle, and R is the runup [Synolakis, *et al.*, 2007].

another analytical solution on a more complex geometry must be verified. The primary focus of this paper is to validate the GPU-SPHysics code using laboratory data provided by NOAA, and follow the standards and procedures for an accurate validation. The inundation

of a solitary wave on a canonical bathymetry and the runup of a tsunami on a conical island will be examined.

CHAPTER II

METHODS

Hardware

To create the highest resolution model possible, and to obtain the greatest accuracy in the tsunami models, only the fastest and most efficient GPUs should be considered. Luckily, the cost of a GPU is significantly less when compared to a cluster of CPUs, which are multiple CPUs ran in parallel [NVIDIA, 2010]. Essentially, a GPU is a small cluster with multiple processors as explained in Chapter I. The GPUs used, to maximize computing power, in our GPU-SPHysics experiments are four of NVIDIAs Tesla C1060 Computing Processors. A Tesla C1060 is shown in figure 5. A single Tesla C1060 produces a teraflop

Fig. 5. Tesla GPU. A NVIDIA Tesla C1060 Computing Processor

(one trillion floating point calculations per second) of double-precision floating point calculations [NVIDIA, 2010]. A floating point calculation is essentially a way for computers to represent non-integer numbers in scientific notation as very large or very small numbers [Goldberg, 1991]. According to the IEEE (Institute of Electrical and Electronics Engineers), the double-precision standard format for a floating point calculation operates on 64-bit memory (what a modern computer typically uses), and provides greater speed and more decimal places of accuracy when calculating a number compared to the single precision format [Goldberg, 1991]. Most scientific numerical computations require double

precision accuracy for correct results. Each Tesla GPU also comes equipped with 240 processing cores creating a combined total of 960 processing cores with the four Tesla GPUs in our workstation. Each Tesla GPU also boasts four gigabytes (a measurement for billions of bytes of information storage) of global memory. A single Tesla GPU is capable of replacing a small CPU cluster which is typically around seven feet tall and can cost as much as five million dollars [NVIDIA, 2010]. A Tesla workstation (depending on how expensive the other hardware required to run the system is) typically costs $10,000 to $17,000 for three Tesla GPUs and enough hardware to support their requirements and is the size of a normal desktop computer. A Tesla GPU workstation has the computing power of an entire room of CPU clusters [NVIDIA, 2010]. The GPUs NVIDIA manufactures typically only allow for one per computer, but the Tesla GPU contains a built-in architecture program called Fermi that allows for multiple Tesla GPUs to be ran simultaneously [NVIDIA, 2010]. In the following experiments, a Tesla workstation containing four Tesla GPUs, and the required hardware to support them, is used to create scientifically accurate and visually stunning models. In order to command the GPUs and create the models, certain software must be utilized.

Software

In order to command the GPUs, CUDA is necessary. The CUDA drivers and toolkit software are downloaded free from NVIDIAs website. The CUDA API (application programming interface) provides extra functions as an extension to the programming language C. The gcc compiler is then used to convert the program into binary code, or code the computer reads. The gcc compiler is obtained from Xcode, a developer package that is provided with every Macintosh computer. The rest of the code that runs on the CPU, the C++ API is used to program the other components of the code. OpenGL, a graphical user interface, creates the visualizations and provides a geometrical shape that the SPH boundaries can

follow. OpenGL allows for a manipulation of the problem environment by giving the user simple shapes to build with such as vector, rectangle, circle, cube, cylinder, and sphere shapes. For one of the experiments, a conical frustum was needed and using OpenGL, a new shape can easily be defined and used in the problem. OpenGL also provides color for the shapes used as well as color schemes in fluid particles which change due to fluctuations in velocity, density and pressure. The visualization environment is created using OpenGL, and offers zoom and rotation functions to view the experiment at any perspective.

GPU-SPHysics structure

The GPU-SPHysics code is broken down into multiple classes that each performs specific operations. The CPU part of the program is the Problem class which has the actual experiments written in C++ [Hérault, *et al*., 2009]. To coordinate the SPH calculations from the CPU to the GPU, the ParticleSystem class calls CUDA kernels which handles each SPH calculation [Hérault, *et al*., 2009]. The velocity and position data, forces, timesteps, neighbor location, and Euler calculations are all performed in the ParticleSystem class [Hérault, *et al*., 2009]. The SPH calculations are performed in timesteps which compute all the forces at a certain interval of time, and with an adaptive timestep, the interval is chosen per timestep so that the model remains stable. However, this portion of the code was well built by Hérault and has no need to be changed. The portion of the code that is edited by the user is the Problem class. In the problem class, the simulation parameters, initial conditions for the particles, and run options are listed as virtual functions and the user must define all these parameters in a subclass [Hérault, *et al*., 2009]. The geometry classes used to build simulations are predetermined, and are easily used to fit the need of each experiment. One of the issues with the current state of the preset geometries is the lack of freedom to produce unique shapes for single experiments. This can be overcome by writing new presets, but even creating simple preset shapes can be cumbersome. In

our experiments, the Problem class is the only manipulated portion of the GPU-SPHysics code.

The solitary wave experiment

To validate the GPU-SPHysics code for tsunamis, laboratory experiments must be set up exactly as required by the NOAA benchmarks. The first of these experiments is the solitary wave test on a simple beach. The tests were performed at the California Institute of Technology in Pasedena, California inside a 31.73-m long, 0.6096-m deep, and 0.3997-m wide wave tank filled with water at various depths [Synolakis, *et al*., 2007]. A flat ramp is installed at the end of the tank with a slope of 1:19.85 (about 2.88 degrees), and the toe of the ramp was 14.95-m from the rest position of a piston used to generate waves at the other end of the tank [Synolakis, *et al*., 2007]. Solitary waves are described in the experiment using a maximum wave height h to water depth H ratio (h/H) [Synolakis, *et al*., 2007]. For the beach with a slope of 1:19.85, the waves would break on the beach when $h/H > 0.045$ [Synolakis, *et al*., 2007]. Hence, the most common breaking and non-breaking wave h/H ratios are used to validate the GPU-SPHysics code. The ratios of $h/H = 0.3$ and $h/H = 0.0.185$ are tested. The piston motion to create a solitary wave is derived from the constant-depth solitary wave equation 16, and includes a factor for solitary wave dissipation. Solitary waves inevitably dissipate as water waves, and a truly meaningful model would account for the small dissipation [Synolakis, *et al*., 2007]. Darlymple has found that in order for a solitary wave to be created, the movement of the piston must follow

$$x_i = \frac{h}{Hk}[1 + tanh\,\alpha] \quad where \quad \alpha = kC(t - t_0) - x_{i-1}k + \frac{h}{H} - 0.5 \tag{24}$$

$$and \quad C = \sqrt{g(H+h)} \quad and \quad k = \sqrt{\frac{3h}{4H^3}} \tag{25}$$

where C is the celerity (wave speed) and x_i is the piston displacement at any instance in time $t > t_0$. The piston displacement accurately creates a solitary wave for the solitary wave experiment and the following conical island experiment. Also, just as in equation 16, the only parameters that need to be defined is the maximum wave height h and the water depth H. With OpenGL, a cube function is used to create the wave tank, and the piston, made from a rectangle function, is the width of the tank and is slightly taller than the water depth. The sloping beach is created using a rectangle function. Once the wave tank dimensions and the slope dimensions are input into the problem, the solitary wave experiment is ready to run.

The conical island experiment

On December 12, 1992, a 7.5 magnitude earthquake off of Flores Island, Indonesia, caused a tsunami that was responsible for over 750 deaths on the nearby Babi Island [Briggs, *et al*, 1995]. The catastrophe caused devastating effects to the island and its inhabitants. One of the most unique aspects of the tsunami at Babi Island was the refraction of the leading wave of the tsunami around the island and its convergence on the backside of the island. The convergence caused an extremely high runup on the back side of the island due to the combination of two waves with equal amplitudes and that were in phase. Motivated by the event, the Coastal Engineering Research Center in Vicksburg, Mississippi performed large-scale laboratory experiments in a 30-m wide, 25-m long, and 0.60-m deep wave basin [Synolakis, *et al*., 2008]. These experiments supplied researchers with valuable laboratory benchmark data to test against models and analytical results for the unique run ups around a conical island. For the test, a large conical island was placed in the basin to replicate the conical shaped Babi Island. The dimensions for the island are 7.2-m toe diameter, 2.2-m crest diameter, 0.625-m height, and a 1:4 slope as drawn in figure 6. The experiment was conducted using multiple h/H ratios at two water depths of 0.42-m and 0.32-m. The

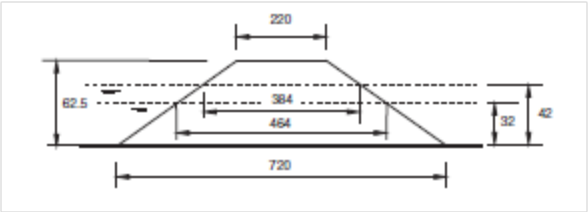

Fig. 6. Island dimensions. A sketch of the conical island used in the experiment. All of the measurements are in centimeters [Synolakis, *et al*., 2007].

0.32-m deep experiment included h/H ratios of 0.045, 0.091, 0.181, and the 0.42-m deep experiment included h/H ratios of 0.046, 0.073, 0.091 [Briggs, *et al*, 1995]. A photograph of the experiment is shown in figure 7. The experiment on GPU-SPHysics for validation includes both depths and multiple ratios tested. With OpenGL, a cube function is used to create the wave tank, and the piston, made from a rectangle function, spans the width of the tank and is slightly taller than the water depth. The piston movement follows equations 24,25 where the maximum wave height h and the water depth H are the only input parameters. A three-dimensional conical frustum file needed to be created in order to simulate the conical island experiment. The radius of the top and bottom as well as the height of the frustum can be changed to fit any experiment's need. The newly developed conical frustum file was added to the GPU-SPHysics package. After the parameters for the experiment are put into the file, the experiment is ready to run.

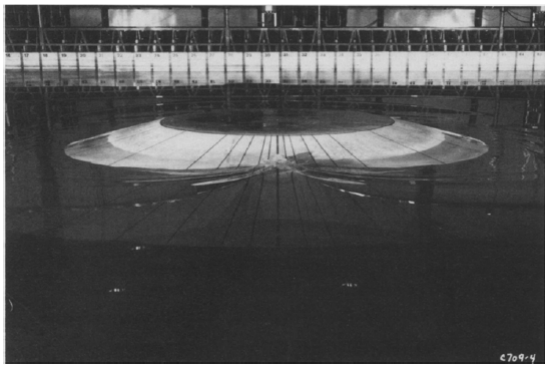

Fig. 7. Conical island laboratory runup. The wave runup as viewed on the backside of the island from the experiments at the Vicksburg, Mississippi Coastal Engineering Research Center. Convergence of the refracted wave around the island is clearly shown [Briggs, *et al*, 1995].

CHAPTER III

RESULTS

Data acquisition

Data from the solitary wave experiment is recorded as either a text (.txt) or a paraview (.pvd) file. Paraview is an application that visualizes three-dimensional parallel data sets. It can create various plots, cross sections, animations and contains many other interpretation tools for a problem set. For SPH, there exists an extension to Paraview called pv-meshless which contains a multitude of extra functions solely for analyzing any type of SPH data. Unfortunately, with GPU-SPHysics data, Paraview is highly unstable and prone to crashing. These effects are highly unpredictable and therefore, Paraview and pv-meshless have been temporarily abandoned until its stability issues can be resolved. Currently, the most useful data collecting method is through a simple visual analysis. However, the visual analysis is very rough and also cannot be applied to every experiment.

Solitary wave results

For the solitary wave experiment, a simple visual analysis and interpretation is sufficient to make a rough estimate of the maximum runup on a gentle sloping beach. By viewing the simulation looking down the vertical axis, the inundation length is easily viewed. By using simple trigonometry, we find the runup, $\Re$, at each recorded time step as a function of the horizontal inundation length, I, still water depth, H, and beach slope angle, β.

$$\Re = I \tan \beta \tag{26}$$

which can be applied to each screenshot of the simulation at a certain time to find the runup. The data is shown in figure 8 where two different model parameters of viscosity and two values for the coefficient of viscosity are compared. Sub-particle scale turbu-

lence (SPS) with laminar viscosity fully represents the turbulence and laminar viscosity in a fluid and the parameter is called SPSVISC [Crespo, 1995a]. Laminar viscosity contains unidirectional stresses[Crespo, 1995a]. SPS viscosity is needed to accurately model the turbulence from a breaking wave. This is especially important when measuring runup since turbulence of a wave break can change the runup. The artificial viscosity parameter (ARTVISC) is a simpler parameter than SPS viscosity, but it does not describe turbulence correctly. The maximum runup for artificial viscosity is 0.151 m, while the maximum runup for SPS viscosity for both coefficients of viscosity is 0.136 m. An analysis of the

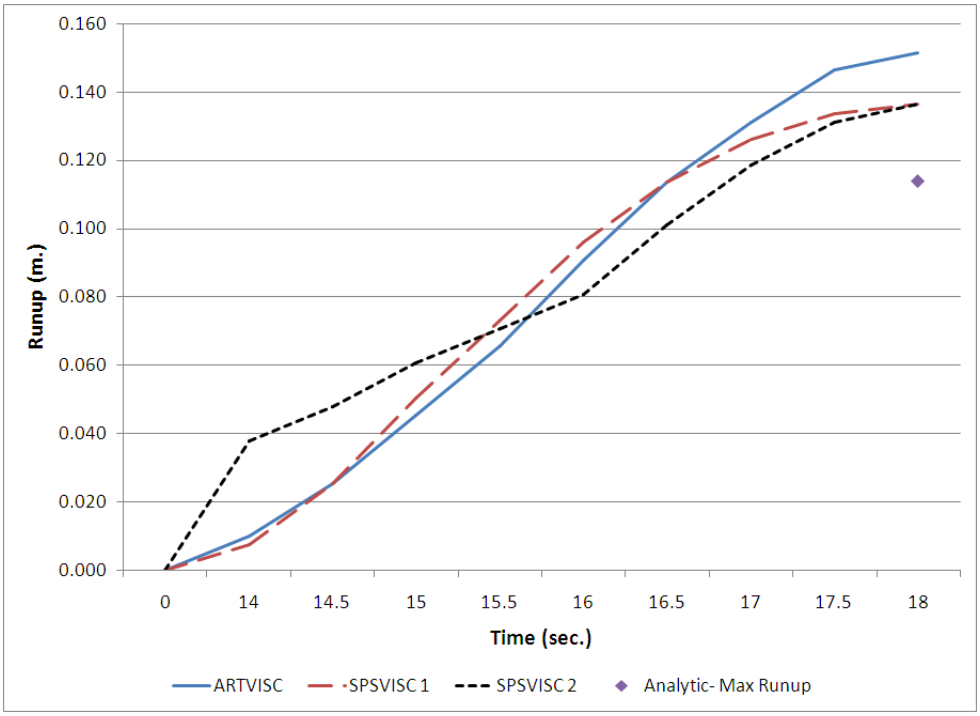

Fig. 8. Runup vs. time. A visual analysis describing the relationship between different solitary wave on a sloping beach runs using SPSVISC (Sub-particle scale turbulence with laminar viscosity), ARTVISC (Artificial Viscosity), and the analytic solution for the maximum runup as described in equation 27. SPSVISC 1 has a viscosity coefficient of 0.05 $m^2\ s^{-1}$ while SPSVISC 2 has a viscosity coefficient of $10^{-6}\ m^2\ s^{-1}$ which is the kinematic viscosity of water at about 20°C.

wave height versus time throughout the inundation is important for a validation. Just as the runup calculation, a visual analysis is the current best possible data collection method, but the deviation of the measurements exceeds the measured values. Figure 9 illustrates a

breaking wave during the inundation of a solitary wave on a sloping beach.

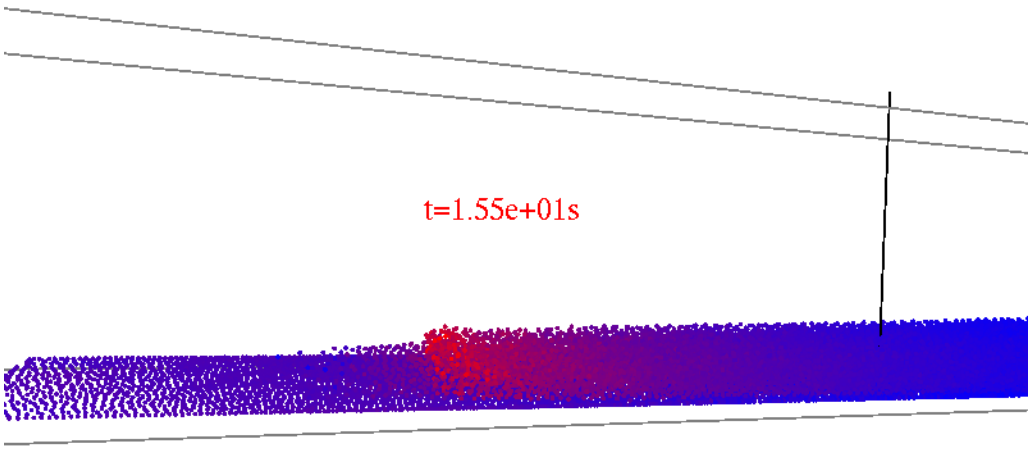

Fig. 9. Solitary wave break. A solitary wave breaking on a sloping beach which was ran with a resolution of 1.2 million particles. Red indicated particles of higher velocities while blue indicates particles of slower velocities. The one meter black, vertical line is placed for scale

Conical island results

The Conical Island experiment requires that runup data be collected at points all around the cone. This is a two-dimensional problem that is highly complex and requires an extrapolation of data to determine the runups around the cone. The maximum runup is given at the 180° transect of the cone as seen in the lab by figure 7 and in GPU-SPHysics by figure 10. Comparing the images, the experiments have a strong visual correlation. In order to further verify the data, the data from the conical island experiment from the coastal engineering research center needs to be directly compared to the data from the GPU-SPHysics

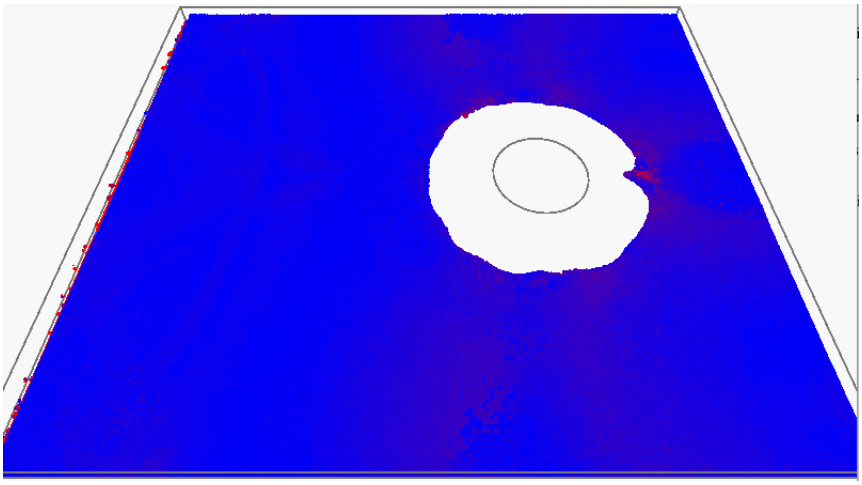

Fig. 10. Conical island GPU-SPHysics maximum runup. The wave runup as viewed on the backside of the island from the Conical Island simulation in GPU-SPHysics. Blue indicates slower fluid velocities while red indicates higher fluid velocities.

conical island simulation in order to fully validate the experiment.

CHAPTER IV

ANALYSIS AND SUMMARY

We use the analytic equation for maximum runup of a solitary wave on a sloping beach as given by Synolakis, *et al*. [2008]

$$\Re_{max} = 2.831\sqrt{\cot\beta}H^{5/4} \tag{27}$$

where $\Re_{max}$ is the maximum runup, β is the slope angle, and H is the maximum wave height. When the parameters for the solitary wave experiment are put into the maximum runup equation, the results for $H/d = 0.3$ correlate very well with the simulated runup in the GPU-SPHysics experiment. The SPS viscosity is closer to the analytic runup than the artificial viscosity. However, the differences between SPS viscosity coefficients surprisingly give the same maximum runup but different inundation velocities. The SPS viscosity that has its coefficient of viscosity equal to that of water simulates the most realistic experiment. Artificial viscosity overestimates the runup since it does not calculate turbulence of the breaking wave which would slow down the inundation and decrease the runup. Due to the inaccuracy of visual measurements, the runup values have a deviation of ± 0.01 meters accuracy since the inundation length was measured using intervals of 0.05 meters. Figure 11 shows an example of the method used to find the inundation length of a solitary wave experiment. With a more in depth analysis and data extrapolation of the experiment, even more accurate results need to be obtained. The lack to tools for extrapolation is the current problematic issue. Using the data files from GPU-SPHysics simulations, the data can be visualized and analyzed using an alternative application to Paraview.

Conclusions

GPU-SPHysics is a viable three-dimensional hydrodynamics modeling code. Utilizing the power of the GPU, GPU-SPHysics has the ability to calculate high resolution N-body

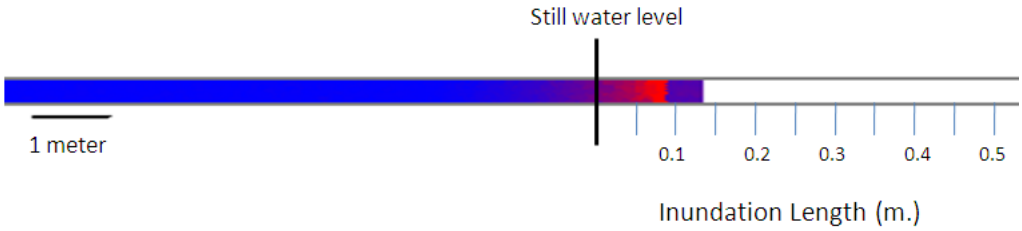

Fig. 11. Inundation length measurement method. As viewed from looking down the vertical axis, so the inundation length is visible. The intervals are of 0.05 meters, and the still water level is marked with the large black line. Red indicates higher fluid velocity while blue indicates slower fluid velocity.

problems with more speed and accuracy than CPU based SPHysics. A valuable application of GPU-SPHysics is to model three-dimensional tsunami inundations. In order to be considered an accurate tsunami inundation model, GPU-SPHysics must be validated according to the model benchmarks provided by NOAA. The analytic, laboratory and field benchmark data must be compared to the results given by respective GPU-SPHysics simulations to validate the models. The solitary wave inundation model on a canonical bathymetry was tested and validated visually and proven to be accurate compared to the analytic solution for maximum runup of a solitary wave. However, data from the model must be compared to experimental data to further validate the simulation. The conical island problem is another useful benchmark used to test the inundation on a complex geometry, and already has shown to have a similar maximum runup to the laboratory experiment. GPU-SPHysics shows great potential to accurately model three-dimensional tsunami inundations and to be used as a hazard analysis tool.

REFERENCES

Briggs, M., *et al*. [1995] "Laboratory Experiments of Tsunami Runup on a Circular Island," *PAGEOPH*, Vol. 3 **4**

Cap, F. [2006] "tsunamis and hurricanes: a mathematical approach," SpringerWien-NewYork, Germany.

Crespo, A.[2008]. "Application of the Smoothed Particle Hydrodynamics model SPHysics to free-surface hydrodynamics." PhD. thesis, Universidade de Vigo, Italy.

Dalrymple, R.A. & Rodgers, B.D. [2005]. Numerical modeling of water waves with the SPH method. *Coastal Engineering*, Vol. 53, pp. 141–147.

Dalrymple, R.A. & Hérault, A. [2009] "Levee Breaching with GPU-SPHysics Code," *Proc. Fourth Workshop, SPHERIC*. ERCOFTAC, Nantes.

Denker, J. [2003]. "Euler's Equation, momentum flow and force-density in fluid dynamics," http://www.av8n.com/physics/euler-flow.htm

Eilbeck, C. [2007] "John Scott Russell and the solitary wave," www.ma.hw.ac.uk/ chris/ scott_russell.html

Goldberg, D. [1991] "What Every Computer Scientist Should Know About Floating-Point Arithmetic," *ACM Computing Surveys*, Vol. 23 **1**

Haberman, W.L.& John, J.E.A. [2006] "Introduction to Fluid Dynamics," Prentice Hall.

Halfhill, T.R. [2008] "Parallel Processing with CUDA," *Microprocessor Report*.

Hérault, A., *et al*. [2009] "Modeling Water Waves in the Surf Zone with GPU-SPHysics," *Proc. Fourth Workshop, SPHERIC*, ERCOFTAC, Nantes.

Kolb, A. & Cuntz, N. [2005] "Dynamic particle coupling for GPU-based fluid simulation," *Proc. 18th Symposium on Simulation Technique*, pp. 722–727.

Madsen, P. A., *et al*. [2008] "On the solitary wave paradigm for tsunamis," *J. Geophys. Res.113*.

Gingold, R.A.& Monaghan, J.J. [1977]. "Smoothed Particle Hydrodynamics: Theory and Application to Non-spherical Stars," *Monthly Notices of the Royal Astronomical Society,* Vol. 181, pp. 375–389.

Monaghan, J.J. [1988]. "An Introduction to SPH," *Computer Physics Communications*, Vol. 48 pp. 89–96.

NVIDIA [2010] "Tesla Computing Solutions," http://www.nvidia.com/object/tesla_computing_solutions.html

Nguyen, H. [2008]. "GPU Gems 3," Pearson Education, Inc., Boston, MA.

Synolakis, C.E. [1987] "The run-up of solitary waves," *Journal of Fluid Mechanics*, Vol. 185, pp. 523–545.

Kanoglu, U. & Synolakis, C.E. [1998] "Long wave runup on piecewise linear topographies," *Journal of Fluid Mechanics*, Vol. 374, pp. 1–28.

Synolakis, C.E. [2007] "Standards, criteria, and procedures for NOAA evaluation of tsunami numerical models," *NOAA Tech. Memo.*, OAR PMEL-135, 55 pp.

Synolakis, C.E. [2008] "Validation and verification of tsunami numerical models," *Pure Applied Geophysics*, Vol. 165, pp. 2197–2228.

Ward, S.N. [2003] "Tsunami Theory-a la Ward," http://es.ucsc.edu/ ward/papers/Basic.pdf

White, F. [2003] "Fluid Mechanics," ed. 5, McGraw-Hill, New York, NY.

CONTACT INFORMATION

Name:	Andrew James Munoz
Professional Address:	c/o Dr. Robert Weiss Department of Geology and Geophysics MS 3115 Texas A&M University College Station, TX 77843
Email Address:	andrewmunoz@ tamu.edu
Education:	B.S., Geophysics, Texas A&M University, Dec. 2010 Suma Cum Laude Undergraduate Research Scholar Phi Kappa Phi